

# A Linear Time Natural Evolution Strategy for Non-Separable Functions

Yi Sun, Faustino Gomez, Tom Schaul, and Jürgen Schmidhuber

IDSIA, University of Lugano & SUPSI, Galleria 2, Manno, CH-6928, Switzerland  
`{yi,tino,tom,juergen}@idsia.ch`

**Abstract.** We present a novel Natural Evolution Strategy (NES) variant, the Rank-One NES (R1-NES), which uses a low rank approximation of the search distribution covariance matrix. The algorithm allows computation of the natural gradient with cost linear in the dimensionality of the parameter space, and excels in solving high-dimensional non-separable problems, including the best result to date on the Rosenbrock function (512 dimensions).

## 1 Introduction

Black-box optimization (also called zero-order optimization) methods have received a great deal of attention in recent years due to their broad applicability to real world problems [7, 9–11, 15, 18]. When the structure of the objective function is unknown or too complex to model directly, or when gradient information is unavailable or unreliable, such methods are often seen as the last resort because all they require is that the objective function can be evaluated at specific points.

In continuous black-box optimization problems, the state-of-the-art algorithms, such as xNES [4] and CMA-ES [8], are all based the same principle [1, 4]: a Gaussian search distribution is repeatedly updated based on the objective function values of sampled points. Usually the full covariance matrix of the distribution is updated, allowing the algorithm to adapt the size and shape of the Gaussian to the local characteristics of the objective function. Full parameterization also provides invariance under affine transformations of the coordinate system, so that ill-shaped, highly non-separable problems can be tackled. However, this generality comes with a price. The number of parameters scales quadratically in the number of dimensions, and the computational cost per sample is at least quadratic in the number of dimensions [13], and sometimes cubic [4, 16].

This cost is often justified since evaluating the objective function can dominate the computation, and thus the main focus is on the improvement of sampling efficiency. However, there are many problems, such as optimizing weights in neural networks where the dimensionality of the parameter space can be very large (e.g. many thousands of weights), the quadratic cost of updating the search distribution can become the computational bottleneck. One possible remedy is to restrict the covariance matrix to be diagonal [14], which reduces the computation per function evaluation to  $O(d)$ , linear in the number  $d$  of dimensions.

Unfortunately, this “diagonal” approach performs poorly when the problem is non-separable because the search distribution cannot follow directions that are not parallel to the current coordinate axes.

In this paper, we propose a new variant of the natural evolution strategy family [17], termed Rank One NES (R1-NES). This algorithm stays within the general NES framework in that the search distribution is adjusted according to the natural gradient [2], but it uses a novel parameterization of the covariance matrix,

$$C = \sigma^2 (I + uu^\top),$$

where  $u$  and  $\sigma$  are the parameters to be adjusted. This parameterization allows the predominant eigen direction,  $u$ , of  $C$  to be aligned in any direction, enabling the algorithm to tackle highly non-separable problems while maintaining only  $O(d)$  parameters. We show through rigorous derivation that the natural gradient can also be effectively computed in  $O(d)$  per sample. R1-NES scales well to high dimensions, and dramatically outperforms diagonal covariance matrix algorithms on non-separable objective functions. As an example, R1-NES reliably solves the non-convex Rosenbrock function up to 512 dimensions.

The rest of the paper is organized as follows. Section 2, briefly reviews the NES framework. The derivation of R1-NES is presented in Section 3. Section 4 empirically evaluates the algorithm on standard benchmark functions, and Section 5 concludes the paper.

## 2 The NES framework

Natural evolution strategies (NES) are a class of evolutionary algorithms for real-valued optimization that maintain a search distribution, and adapt the distribution parameters by following the *natural* gradient of the expected function value. The success of such algorithms is largely attributed to the use of natural gradient, which has the advantage of always pointing in the direction of the steepest ascent, even if the parameter space is not Euclidean. Moreover, compared to regular gradient, natural gradient reduces the weights of gradient components with higher uncertainty, therefore making the update more reliable. As a consequence, NES algorithms can effectively cope with objective functions with ill-shaped landscapes, especially preventing premature convergence on plateaus and avoiding overaggressive steps on ridges [16].

The general framework of NES is given as follows: At each time step, the algorithm samples  $n \in \mathbb{N}$  new samples  $x_1, \dots, x_n \sim \pi(\cdot|\theta)$ , with  $\pi(\cdot|\theta)$  being the search distribution parameterized by  $\theta$ . Let  $f: \mathbb{R}^d \mapsto \mathbb{R}$  be the objective function to maximize. The expected function value under the search distribution is

$$J(\theta) = \mathbb{E}_\theta[f(x)] = \int f(x) \pi(x|\theta) dx.$$

Using the log-likelihood trick, the gradient w.r.t. the parameters can be written as

$$\begin{aligned}\nabla_{\theta} J &= \nabla_{\theta} \int f(x) \pi(x|\theta) dx \\ &= \int f(x) \pi(x|\theta) \frac{\nabla_{\theta} \pi(x|\theta)}{\pi(x|\theta)} dx \\ &= E[f(x) \nabla_{\theta} \log \pi(x|\theta)],\end{aligned}$$

from which we obtain the Monte-Carlo estimate

$$\nabla_{\theta} J \simeq \frac{1}{n} \sum_{i=1}^n f(x_i) \nabla_{\theta} \log \pi(x_i|\theta).$$

of the search gradient. The key step of NES then consists of replacing this gradient by the natural gradient

$$\tilde{\nabla}_{\theta} J = F^{-1} \nabla_{\theta} J,$$

where

$$F = E \left[ \nabla_{\theta} \log \pi(x_i|\theta) \nabla_{\theta} \log \pi(x_i|\theta)^{\top} \right]$$

is the Fisher information matrix (See Fig.1 for an illustration). This leads to a straightforward scheme of natural gradient ascent for iteratively updating the parameters

$$\begin{aligned}\theta &\leftarrow \theta + \eta \tilde{\nabla}_{\theta} J \\ &= \theta + \frac{\eta}{n} \sum_{i=1}^n f(x_i) F^{-1} \nabla_{\theta} \log \pi(x_i|\theta) \\ &= \theta + \frac{\eta}{n} \sum_{i=1}^n f(x_i) \tilde{\nabla}_{\theta} \log \pi(x_i|\theta).\end{aligned}\tag{1}$$

The sequence of 1) sampling an offspring population, 2) computing the corresponding Monte Carlo estimate of the gradient, 3) transforming it into the natural gradient, and 4) updating the search distribution, constitutes one iteration of NES.

The most difficult step in NES is the computation of the Fisher information matrix with respect to the parameterization. For full Gaussian distribution, the Fisher can be derived analytically [4,16]. However, for arbitrary parameterization of  $C$ , the Fisher matrix can be highly non-trivial.

### 3 Natural gradient of the rank-one covariance matrix approximation

In this paper, we consider a special formulation of the covariance matrix

$$C = \sigma^2 (I + uu^{\top}),$$

with parameter set  $\theta = \langle \sigma, u \rangle$ . The special part of the parameterization is the vector  $u \in \mathbb{R}^d$ , which corresponds to the predominant direction of  $C$ . This allows the search distribution to be aligned in any direction by adjusting  $u$ , enabling the algorithm to follow valleys not aligned with the current coordinate axes, which is essential for solving non-separable problems.

Since  $\sigma$  should always be positive, following the same procedure in [4], we parameterize  $\sigma = e^\lambda$ , so that  $\lambda \in \mathbb{R}$  can be adjusted freely using gradient descent without worrying about  $\sigma$  becoming negative. The parameter set is adjusted to  $\theta = \langle \lambda, u \rangle$  accordingly.

From the derivation of [16], the natural gradient on the sample mean is given by

$$\tilde{\nabla}_\mu \log p(x|\theta) = x - \mu. \quad (2)$$

In the subsequent discussion we always assume  $\mu = 0$  for simplicity. It is straightforward to sample from  $\mathcal{N}(0, C)$ <sup>1</sup> by letting  $y \sim \mathcal{N}(0, I)$ ,  $z \sim \mathcal{N}(0, 1)$ , then

$$x = \sigma(y + zu) \sim \mathcal{N}(0, C).$$

The inverse of  $C$  can also be computed easily as

$$C^{-1} = \sigma^{-2} \left( I - \frac{1}{1 + r^2} uu^\top \right),$$

where  $r^2 = u^\top u$ . Using the relation  $\det(I + uu^\top) = 1 + u^\top u$ , the determinant of  $C$  is

$$|C| = \sigma^{2d} (1 + r^2).$$

Knowing  $C^{-1}$  and  $|C|$  allows the log-likelihood to be written explicitly as

$$\begin{aligned} \log p(x|\theta) &= \text{const} - \frac{1}{2} \log |C| - \frac{1}{2} x^\top C^{-1} x \\ &= \text{const} - \lambda d - \frac{1}{2} \log(1 + r^2) - \frac{1}{2} e^{-2\lambda} x^\top x + \frac{1}{2} \frac{e^{-2\lambda}}{1 + r^2} (x^\top u)^2. \end{aligned}$$

The regular gradient with respect to  $\lambda$  and  $u$  can then be computed as:

$$\nabla_\lambda \log p(x|\theta) = -d + e^{-2\lambda} \left( x^\top x - \frac{(x^\top u)^2}{1 + r^2} \right), \quad (3)$$

$$\nabla_u \log p(x|\theta) = -\frac{u}{1 + r^2} + e^{-2\lambda} \left[ -\frac{(x^\top u)^2 u}{(1 + r^2)^2} + \frac{(x^\top u) x}{1 + r^2} \right]. \quad (4)$$

Replacing  $x$  with  $e^\lambda(y + zu)$ , then the Fisher can be computed by marginalizing out i.i.d. standard Gaussian variables  $y$  and  $z$ , namely,

$$\begin{aligned} F &= E_x \left[ \nabla_\theta \log p(x|\theta) \nabla_\theta \log p(x|\theta)^\top \right] \\ &= E_{y,z} \left[ \nabla_\theta \log p(y + zu|\theta) \nabla_\theta \log p(y + zu|\theta)^\top \right]. \end{aligned}$$

---

<sup>1</sup> For succinctness, we always assume the mean of the search distribution is 0. This can be achieved easily by shifting the coordinates.

Since elements in  $\nabla_{\theta} \log p(x|\theta) \nabla_{\theta} \log p(x|\theta)^{\top}$  are essentially polynomials of  $y$  and  $z$ , their expectations can be computed analytically<sup>2</sup>, which gives the exact Fisher information matrix

$$F = \begin{bmatrix} 2d & \frac{2u^{\top}}{1+r^2} \\ \frac{2u}{1+r^2} & B \end{bmatrix},$$

with

$$B = \frac{1}{1+r^2} \left[ r^2 I + \frac{1-r^2}{1+r^2} uu^{\top} \right].$$

Let  $v = u/r$ , then

$$F = \frac{r^2}{1+r^2} \begin{bmatrix} 2d \frac{1+r^2}{r^2} & 2 \frac{v^{\top}}{r} \\ 2 \frac{v}{r} & I + \frac{1-r^2}{1+r^2} vv^{\top} \end{bmatrix}.$$

The inverse of  $F$  is thus given by

$$F^{-} = \frac{1+r^2}{r^2} \begin{bmatrix} 2d \frac{1+r^2}{r^2} & 2 \frac{v^{\top}}{r} \\ 2 \frac{v}{r} & I + \frac{1-r^2}{1+r^2} vv^{\top} \end{bmatrix}^{-}.$$

We apply the formula for block matrix inverse in [12]

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^{-} = \begin{bmatrix} C_1^{-} & -A_{11}^{-} A_{12} C_2^{-} \\ -C_2^{-} A_{21} A_{11}^{-} & C_2^{-} \end{bmatrix},$$

where  $C_1 = A_{11} - A_{12} A_{22}^{-} A_{21}$ , and  $C_2 = A_{22} - A_{21} A_{11}^{-} A_{12}$  are the Schur complements. Let  $F$  be partitioned as above, then

$$B^{-} = I - \frac{1-r^2}{2} vv^{\top},$$

and the Shur complements are

$$\begin{aligned} C_1 &= 2d \frac{1+r^2}{r^2} - 4 \frac{v^{\top}}{r} \left( I - \frac{1-r^2}{2} vv^{\top} \right) \frac{v}{r} \\ &= 2d \left( \frac{1+r^2}{r^2} \right) - 2 \frac{1+r^2}{r^2} \\ &= 2(d-1) \left( \frac{1+r^2}{r^2} \right), \end{aligned}$$

and

$$\begin{aligned} C_2 &= I + \frac{1-r^2}{1+r^2} vv^{\top} - \frac{2vv^{\top}}{d(1+r^2)} \\ &= I + \frac{1}{1+r^2} \left[ 1-r^2 - \frac{2}{d} \right] vv^{\top}, \end{aligned}$$

---

<sup>2</sup> The derivation is tedious, thus omitted here. All derivations are numerically verified using Monte-Carlo simulation.

whose inverse is given by

$$C_2^- = I + \frac{2 + d(r^2 - 1)}{2(d - 1)} vv^\top.$$

Combining the results gives the analytical solution of the inverse Fisher:

$$F^- = \frac{1 + r^2}{2r^2(d - 1)} \begin{bmatrix} \frac{r^2}{1+r^2} & -rv^\top \\ -rv & I + [2 + d(r^2 - 1)] vv^\top \end{bmatrix}.$$

Multiplying  $F^-$  with the regular gradient in Eq.3 and Eq.4 gives the natural gradient for  $\lambda$  and  $u$ :

$$\tilde{\nabla}_\lambda \log p(x|\theta) = \frac{1}{2(d - 1)} \left[ (e^{-2\lambda} x^\top x - d) - (e^{-2\lambda} (x^\top v)^2 - 1) \right]. \quad (5)$$

and

$$\tilde{\nabla}_u \log p(x|\theta) = \frac{e^{-2\lambda}}{2(d - 1)r} \left[ (1 - d) (x^\top v)^2 + (r^2 + 1) \left( (x^\top v)^2 - x^\top x \right) \right]. \quad (6)$$

Note that computing both  $\tilde{\nabla}_\lambda \log p(x|\theta)$  and  $\tilde{\nabla}_u \log p(x|\theta)$  requires only the inner products  $x^\top x$  and  $x^\top v$ , therefore can be done  $O(d)$  storage and time.

### 3.1 Reparameterization

The natural gradient above is obtained with respect to  $u$ . However, direct gradient update on  $u$  has an unpleasant property when  $\tilde{\nabla}_u \log p(x|\theta)$  is in the opposite direction of  $u$ , which is illustrated in Fig. 2(a). In this case, the gradient tends to shrink  $u$ . However, if  $\tilde{\nabla}_u \log p(x|\theta)$  is large, adding the gradient will flip the direction of  $u$ , and the length of  $u$  might even grow. This causes numerical problems, especially when  $r$  is small. A remedy is to separate the length and direction of  $u$ , namely, reparameterize  $u = e^c v$ , where  $\|v\| = 1$  and  $e^c$  is the length of  $u$ . Then the gradient update on  $c$  will never flip  $u$ , and thus avoid the problem.

Note that for small change  $\delta u$ , the update on  $c$  and  $v$  can be obtained from

$$\begin{aligned} \delta c &= \frac{1}{2} \log (u + \delta u)^\top (u + \delta u) - c \\ &\simeq \frac{1}{2} \log (u^\top u + 2\delta u^\top u) - c \\ &= \frac{1}{2} \log u^\top u + \frac{1}{2} \log \left( 1 + \frac{2\delta u^\top u}{u^\top u} \right) - c \\ &\simeq \frac{\delta u^\top u}{u^\top u} \end{aligned}$$

---

**Algorithm 1: R1-NES( $n$ )**


---

```

1 while not terminate do
2   for  $i = 1$  to  $n$  do
3      $y_i \leftarrow \mathcal{N}(0, I)$ 
4      $z_i \leftarrow \mathcal{N}(0, 1)$ 
5      $x_i \leftarrow e^\lambda (y_i + z_i u)$  //generate sample
6      $\text{fitness}[i] \leftarrow f(\mu + x_i)$ 
7   end
8   Compute the natural gradient for  $\mu$ ,  $\lambda$ ,  $u$ ,  $c$ , and  $v$  according to Eq.2, 5, 6,
   7, and 8, and combine them using Eq.1
9    $\mu \leftarrow \mu + \eta \tilde{\nabla}_\mu J$ 
10   $\lambda \leftarrow \lambda + \eta \tilde{\nabla}_\lambda J$ 
11  if  $\tilde{\nabla}_c \log p(x|\theta) < 0$  then
12     $c \leftarrow c + \eta \tilde{\nabla}_c J$ 
13     $v \leftarrow \frac{v + \eta \tilde{\nabla}_v J}{\|v + \eta \tilde{\nabla}_v J\|}$ 
14     $u \leftarrow e^c v$ 
15  else
16     $u \leftarrow u + \eta \tilde{\nabla}_u J$  //additive update
17     $c \leftarrow \log \|u\|$ 
18     $v \leftarrow \frac{u}{\|u\|}$ 
19  end
20 end

```

---

and

$$\begin{aligned}
\delta v &= \frac{u + \delta u}{\sqrt{(u + \delta u)^\top (u + \delta u)}} - v \\
&\simeq \frac{u + \delta u}{(u^\top u + 2\delta u^\top u)^{\frac{1}{2}}} - \frac{u}{(u^\top u)^{\frac{1}{2}}} \\
&= \frac{u + \delta u}{(u^\top u)^{\frac{1}{2}} \left(1 + \frac{2\delta u^\top u}{u^\top u}\right)^{\frac{1}{2}}} - \frac{u}{(u^\top u)^{\frac{1}{2}}} \\
&\simeq \frac{(u + \delta u) \left(1 - \frac{\delta u^\top u}{u^\top u}\right)}{(u^\top u)^{\frac{1}{2}}} - \frac{u}{(u^\top u)^{\frac{1}{2}}} \\
&= \frac{1}{(u^\top u)^{\frac{1}{2}}} \left[ \delta u - \frac{\delta u^\top u}{u^\top u} u \right].
\end{aligned}$$

The natural gradient on  $c$  and  $v$  is given by letting  $\delta u \propto \tilde{\nabla}_u \log p(x|\theta)$ , thanks to the invariance property:

$$\tilde{\nabla}_c \log p(x|\theta) = r^{-1} \tilde{\nabla}_u \log p(x|\theta)^\top v \quad (7)$$

$$\tilde{\nabla}_v \log p(x|\theta) = r^{-1} \left[ \tilde{\nabla}_u \log p(x|\theta) - \left( \tilde{\nabla}_u \log p(x|\theta)^\top v \right) v \right], \quad (8)$$

Note that computing  $\tilde{\nabla}_c \log p(x|\theta)$  and  $\tilde{\nabla}_v \log p(x|\theta)$  involves only inner products between vectors, which can also be done linearly in the number of dimensions.

Using the parameterization  $\langle c, v \rangle$  introduces another problem. When  $r$  is small,  $\tilde{\nabla}_c \log p(x|\theta)$  tends to be large, and thus directly updating  $c$  causes  $r$  to grow exponentially, resulting in numerical instability, as shown in Fig. 2(b). In this case, the additive update on  $u$ , rather than the update on  $\langle c, v \rangle$  is more stable. In our implementation, the additive update on  $u$  is used if  $\tilde{\nabla}_c \log p(x|\theta) > 0$ , otherwise the update is on  $\langle c, v \rangle$ . This solution proved to be numerically stable in all our tests. Algorithm 1 shows the complete R1-NES algorithm in pseudocode.

## 4 Experiments

The R1-NES algorithm was evaluated on the twelve noise-free unimodal functions [6] in the ‘Black-Box Optimization Benchmarking’ collection (BBOB) from the 2010 GECCO Workshop for Real-Parameter Optimization. In order to make the results comparable those of other methods, the setup in [5] was used, which transforms the pure benchmark functions to make the parameters non-separable (for some) and avoid trivial optima at the origin.

R1-NES was compared to xNES [3], SNES [14] on each benchmark with problem dimensions  $d = 2^k$ ,  $k = \{1..9\}$  (20 runs for each setup), except for xNES, which was only run up  $k = 6, d = 64$ . Note that xNES serves as a proper baseline since it is state-of-the-art, achieving performance on par with the popular CMA-ES. The reference machine is an Intel Core i7 processor with 1.6GHz and 4GB of RAM.

Fig. 3 shows the results for the eight benchmarks on which R1-NES performs at least as well as the other methods, and often much better. For dimensionality under 64, R1-NES is comparable to xNES in terms of the number of fitness evaluations, indicating that the rank-one parameterization of the search distribution effectively captures the local curvature of the fitness function (see Fig. 5 for example). However, the time required to compute the update for the two algorithms differs drastically, as depicted in Fig. 4. For example, a typical run of xNES in 64 dimensions takes hours (hence the truncated xNES curves in all graphs), compared to minutes for R1-NES. As a result, R1-NES can solve these problems up to 512 dimensions in acceptable time. In particular, the result on the 512-dimensional Rosenbrock function is, to our knowledge, the best to date. We estimate that optimizing the 512-dimensional sphere function with xNES (or any other full parameterization method, e.g. CMA-ES) would take over a year in computation time on the same reference hardware. It is also worth pointing out that sNES, though sharing similar, low complexity per function evaluation, can only solve separable problems (Sphere, Linear, AttractiveSector, and Ellipsoid).

Fig. 6 shows four cases (Ellipsoid, StepEllipsoid, RotatedEllipsoid, and Tablet) for which R1-NES is not suited, highlighting a limitation of the algorithm. Three of the four functions are from the Ellipsoid family, where the fitness functions



are variants of the type

$$f(x_1, \dots, x_d) = \sum_{i=1}^d x_i^{2000 \cdot \frac{i-1}{d}}.$$

The eigenvalues of the Hessian span several orders of magnitude, and the parameterization with a single predominant direction is not enough to approximate the Hessian, resulting in poor performance. The other function where R1-NES fails is the Tablet function where all but a one eigendirection has a large eigenvalue. Since the parameterization of R1-NES only allows a single direction to have a large eigenvalue, the shape of the Hessian cannot be effectively approximated.

## 5 Conclusion and future work

We presented a new black-box optimization algorithm R1-NES that employs a novel parameterization of the search distribution covariance matrix which allows a predominant search direction to be adjusted using the natural gradient with complexity linear in the dimensionality. The algorithm shows excellent performance in a number of high-dimensional non-separable problems that, to date, have not been solved with other parameterizations of similar complexity.

Future work will concentrate on overcoming the limitations of the algorithm (shown in Fig 6). In particular, we intend to extend the algorithm to a) incorporate multiple search directions, and b) enable each search direction to *shrink* as well as grow.

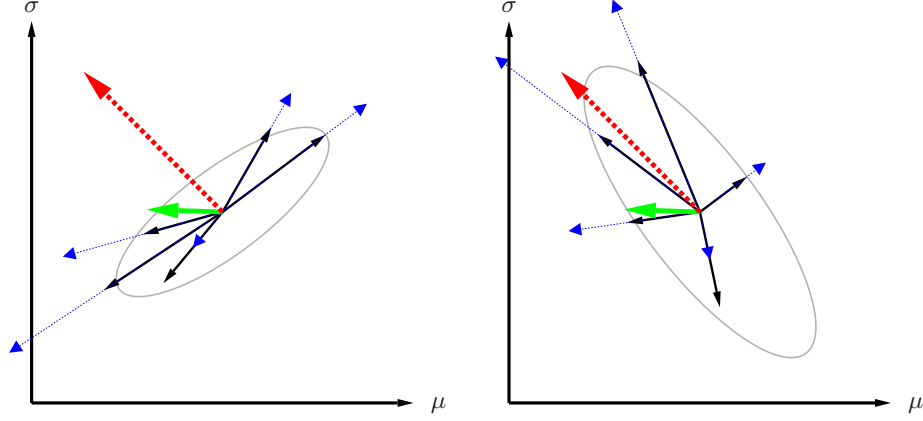
## Acknowledgement

This research was funded in part by Swiss National Science Foundation grants 200020-122124, 200020-125038, and EU IM-CLeVeR project (#231722).

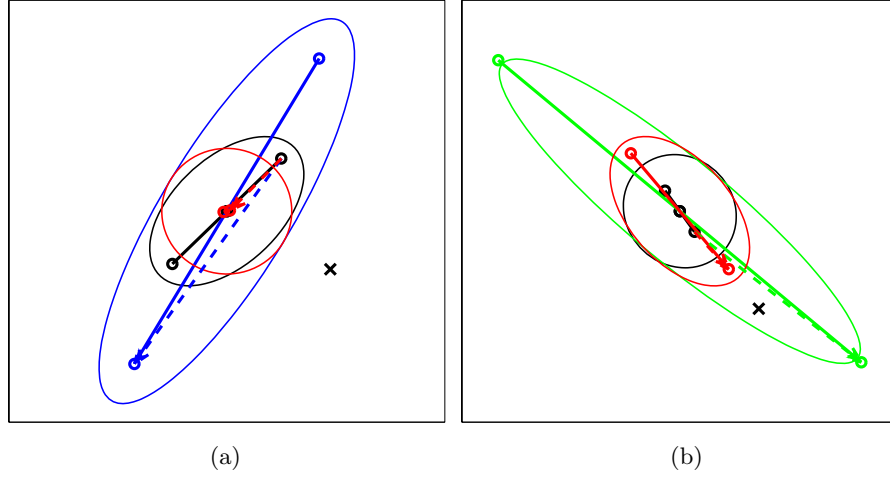
## References

1. Y. Akimoto, Y. Nagata, I. Ono, and S. Kobayashi. Bidirectional Relation between CMA Evolution Strategies and Natural Evolution Strategies. In *Parallel Problem Solving from Nature (PPSN)*, 2010.
2. S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.
3. T. Glasmachers, T. Schaul, Y. Sun, D. Wierstra, and J. Schmidhuber. Exponential Natural Evolution Strategies. In *Genetic and Evolutionary Computation Conference (GECCO)*, Portland, OR, 2010.
4. T. Glasmachers, T. Schaul, Y. Sun, and J. Schmidhuber. Exponential natural evolution strategies. In *GECCO’10*, 2010.
5. N. Hansen and A. Auger. Real-parameter black-box optimization benchmarking 2010: Experimental setup, 2010.

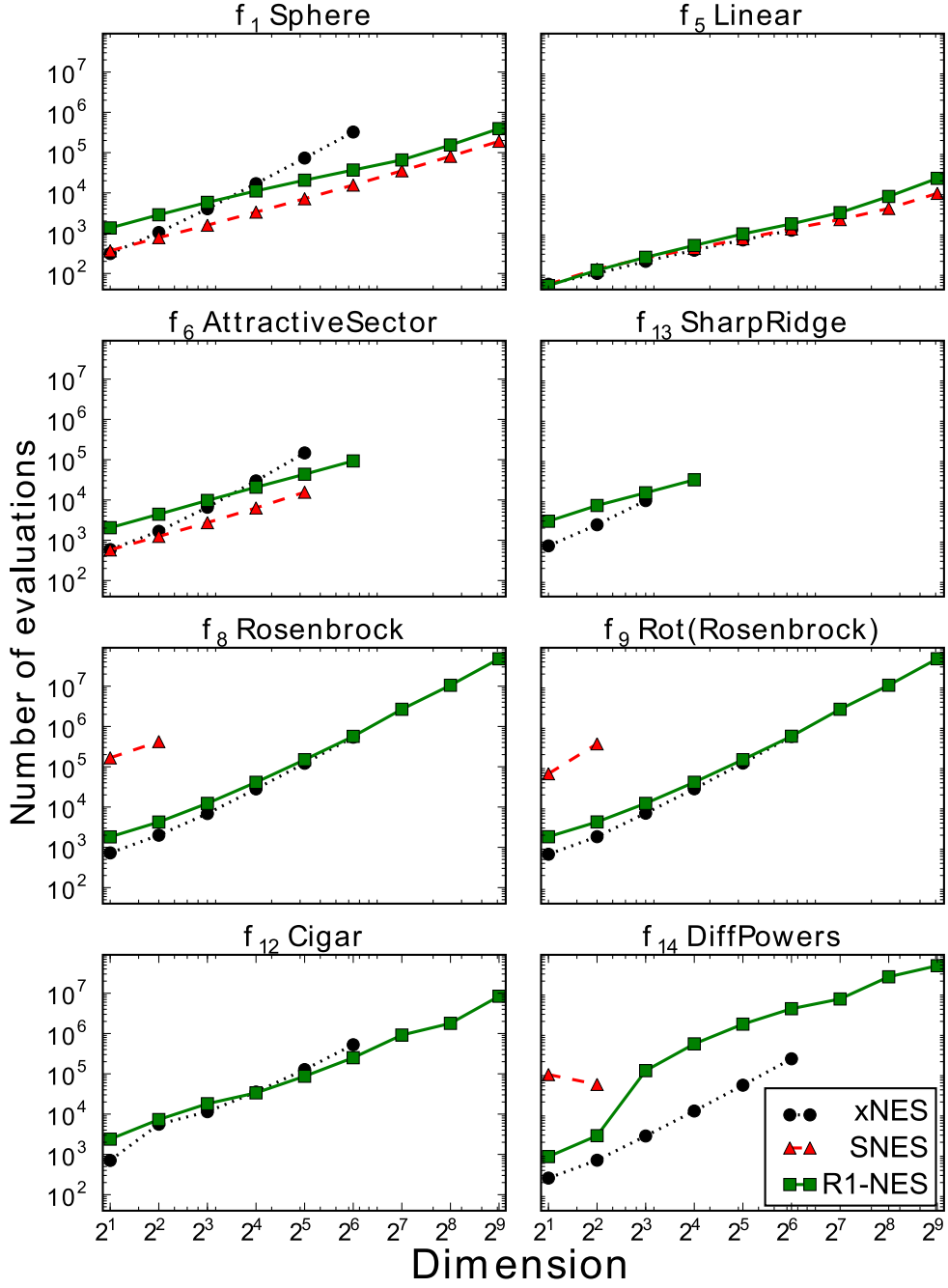
6. N. Hansen and S. Finck. Real-parameter black-box optimization benchmarking 2010: Noiseless functions definitions, 2010.
7. N. Hansen, A. S. P. Niederberger, L. Guzzella, and P. Koumoutsakos. A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. *Trans. Evol. Comp.*, 13:180–197, 2009.
8. N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.
9. M. Hasenjäger, B. Sendhoff, T. Sonoda, and T. Arima. Three dimensional evolutionary aerodynamic design optimization with CMA-ES. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 2173–2180, New York, NY, USA, 2005. ACM.
10. M. Jebalia, A. Auger, M. Schoenauer, F. James, and M. Postel. Identification of the isotherm function in chromatography using cma-es. In *IEEE Congress on Evolutionary Computation*, pages 4289–4296, 2007.
11. J. Klockgether and H. P. Schwefel. Two-phase nozzle and hollow core jet experiments. In *Proc. 11th Symp. Engineering Aspects of Magnetohydrodynamics*, pages 141–148, 1970.
12. K. B. Petersen and M. S. Pedersen. The matrix cookbook, 2008.
13. R. Ros and N. Hansen. A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity. In R. et al., editor, *Parallel Problem Solving from Nature, PPSN X*, pages 296–305. Springer, 2008.
14. T. Schaul, T. Glasmachers, and J. Schmidhuber. High Dimensions and Heavy Tails for Natural Evolution Strategies. In *To appear in: Genetic and Evolutionary Computation Conference (GECCO)*, 2011.
15. O. M. Shir and T. Bäck. The second harmonic generation case-study as a gateway for es to quantum control problems. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 713–721, New York, NY, USA, 2007. ACM.
16. Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber. Stochastic search using the natural gradient. In *ICML'09*, 2009.
17. D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber. Natural Evolution Strategies. In *Proceedings of the Congress on Evolutionary Computation (CEC08), Hongkong*. IEEE Press, 2008.
18. S. Winter, B. Brendel, and C. Igel. Registration of bone structures in 3d ultrasound and ct data: Comparison of different optimization strategies. *International Congress Series*, 1281:242 – 247, 2005.



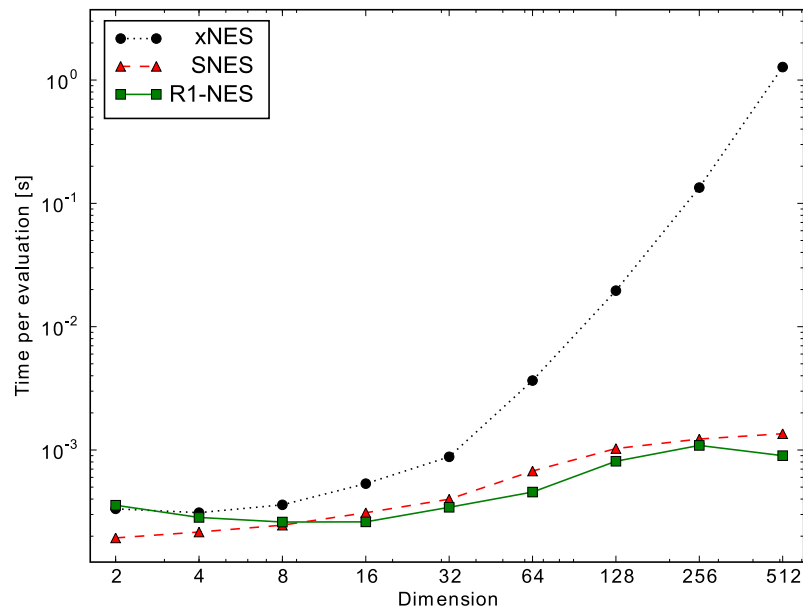
**Fig.1. Plain versus natural gradient in parameter space.** Consider two parameters, e.g.  $\theta = (\mu, \sigma)$ , of the search distribution. On the left, the solid (black) arrows indicate the gradient samples  $\nabla_{\theta} \log \pi(x|\theta)$ , the dotted (blue) arrows correspond to  $f(x) \cdot \nabla_{\theta} \log \pi(x|\theta)$ , that is, the same gradient estimates, but scaled with fitness. Combining these, the bold (green) arrow indicates the (sampled) fitness gradient  $\nabla_{\theta} J$ , while the bold dashed (red) arrow indicates the corresponding natural gradient  $\tilde{\nabla}_{\theta} J = F^{-1} \nabla_{\theta} J$ . Being random variables with expectation zero, the distribution of the black arrows is governed by their covariance (the gray ellipse). Note that this covariance is a quantity in *parameter space* (where the  $\theta$  reside); not to be confused with that of the *search space* (where the samples  $x$  reside). In contrast, on the right, the solid (black) arrows represent  $\tilde{\nabla}_{\theta} \log \pi(x|\theta)$ , and dotted (blue) arrows indicate the *natural* gradient samples  $f(x) \cdot \tilde{\nabla}_{\theta} \log \pi(x|\theta)$ , resulting in the natural gradient (dashed red). The covariance of the solid arrows on the right hand side turns out to be the inverse of the covariance of the solid arrows on the left. This has the effect that when computing the natural gradient, directions with high variance (uncertainty) are penalized and thus shrunk, while components with low variance (high certainty) are boosted, since these components of the gradient samples deserve more trust. This makes the (dashed red) natural gradient a much more trustworthy update direction than the (green) plain gradient.



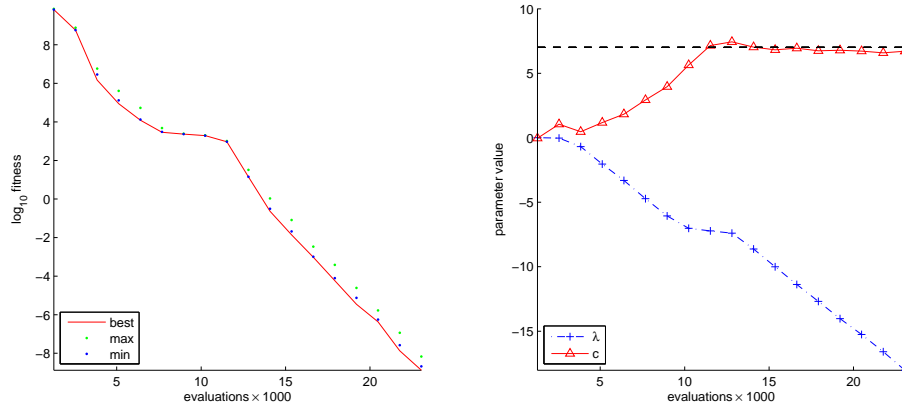
**Fig. 2. Illustration of the change in parameterization.** In both panels, the black lines and ellipses refer to the current predominant direction  $u$ , and the corresponding search distribution from which samples are drawn. The black cross denotes one such sample that is being used to update the distribution. In the left panel, the direction of the selected point is almost perpendicular to  $u$ , resulting in a large gradient, reducing  $u$  (the dotted blue line). However, direct gradient update on  $u$  will flip the direction of  $u$ . As a result,  $u$  stays in the same undesired direction, but with increased length. In contrast, performing update on  $c$  and  $v$  gives the predominant search direction depicted in the red, with  $u$  shrunk properly. The right panel shows another case where the selected point aligns with the search direction, and performing the exponential update on  $c$  and  $v$  causes  $u$  to increase dramatically (green line & ellipsoid). This effect is prevented by performing the additive update (Eq. 6) on  $u$  (red line & ellipsoid).



**Fig. 3. Performance comparison on BBOB unimodal benchmarks.** Log-log plot of the median number of fitness evaluations (over 20 trials) required to reach the target fitness value of  $-10^{-8}$  for unimodal benchmark functions for which R1-NES is well suited, on dimensions 2 to 512 (cases for which 90% or more of the runs converged prematurely are not shown). Note that xNES consistently solves all benchmarks on small dimensions ( $\leq 64$ ), with a scaling factor that is almost the same over all functions.



**Fig.4. Computation time per function evaluation**, for the three algorithms, on problem dimensions ranging from 2 to 512. Both SNES and R1-NES scale linearly, whereas the cost grows cubically for xNES.



**Fig.5. Behavior of R1-NES on the 32-dimensional cigar function:**  $f(x_1, \dots, x_d) = 10^6 x_1^2 + x_2^2 + \dots + x_d^2$ . The left panel shows the best fitness found so far, and the min and max fitness in the current population. The right panel shows how  $\lambda$  and  $c$  evolve over time. Note that the  $\lambda$  decreases almost linearly, indicating that all the other directions except the predominant one shrink exponentially. In contrast,  $c$  first increases, and then stabilizes around  $\log 1000$  (the black line). As a result,  $I + uu^\top$  corresponds to the Hessian of the cigar function  $[10^6, 1, \dots, 1]$ .

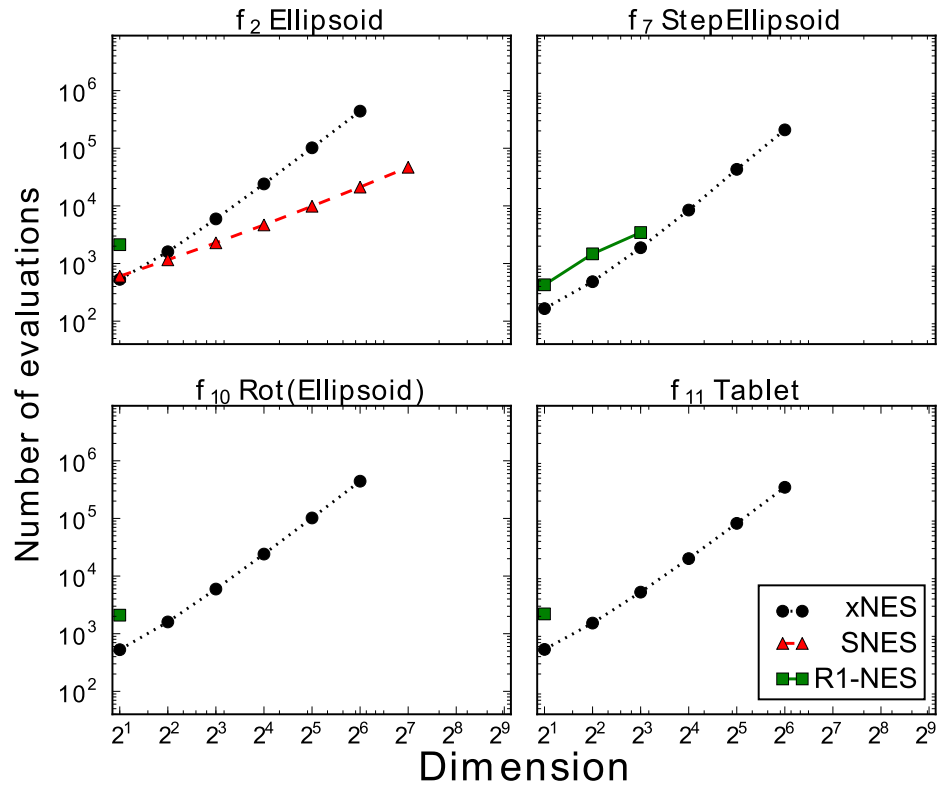


Fig. 6. Performance comparison on BBOB unimodal benchmarks for which R1-NES is not well suited. For these four functions a single eigendirection is not enough. Not that SNES solves the Ellipsoid function because it is separable.